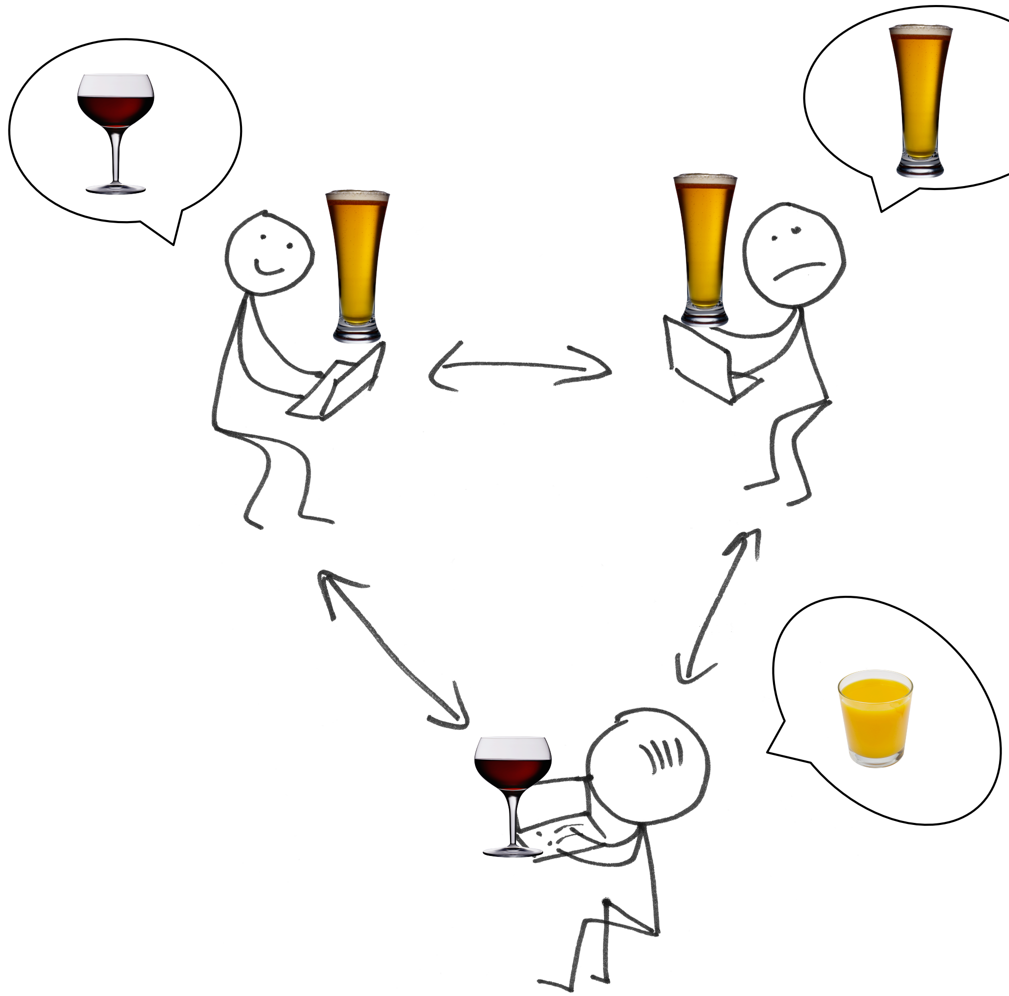
# Wait-Freedom with Advice

Carole Delporte    Hugues Fauconnier
U Paris Diderot

Eli Gafni    Petr Kuznetsov
UCLA    TU Berlin/Telekom Innovation Labs

PODC 2012

# Solving a task: correctness



# Distributed tasks (I,O,Δ)

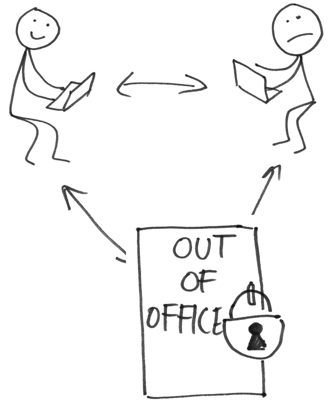
- I – set of input vectors
- O – set of output vectors
- Task specification Δ: $I \rightarrow 2^O$

k-set agreement:

- Processes start with inputs in V (|V|>k)
- The set of outputs is a subset of inputs of size at most k
- k=1: consensus
- Colorless: allows for adopting inputs or outputs

# Solving a task: progress

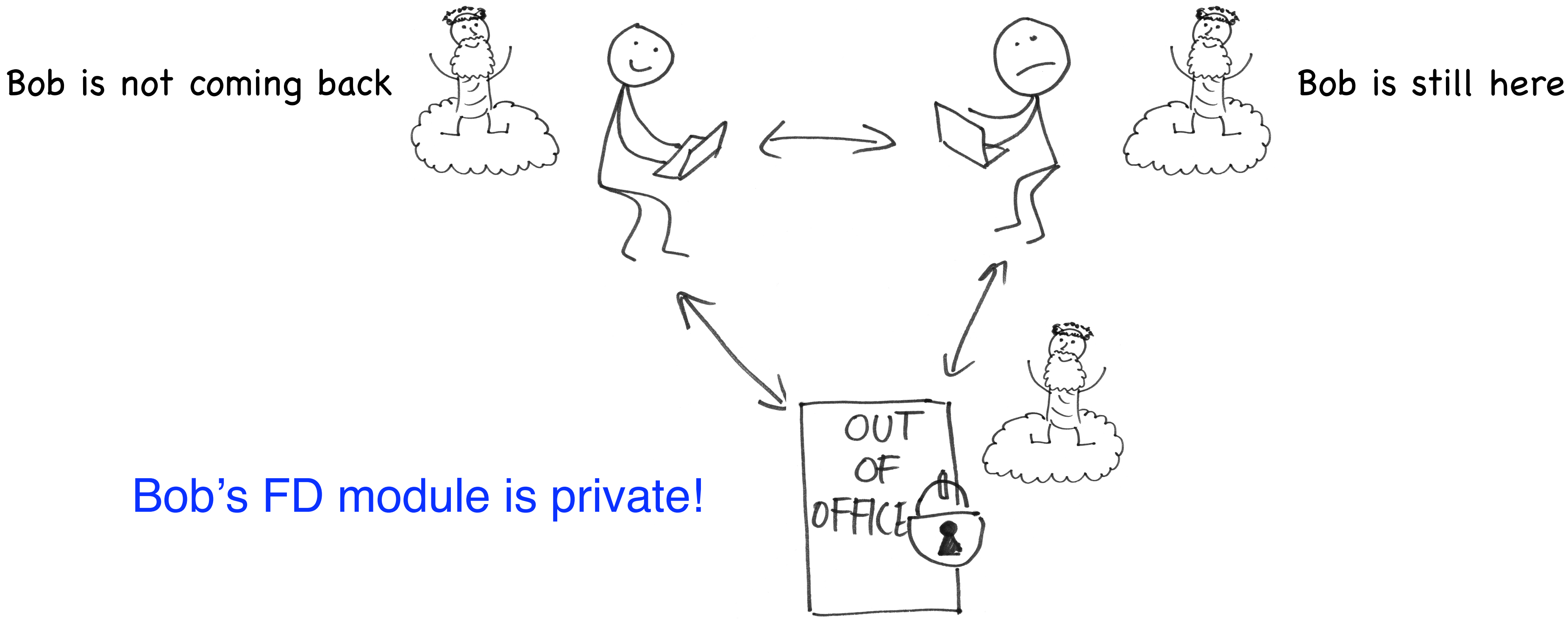
- Every process outputs
  - ✓ Unrealistic for systems with failures or very long delays




- Every process taking enough steps outputs (wait-freedom)
  - ✓ Individual progress is a liveness property: a slow process may wake up and make progress later
  - ✓ No notion of failures

# But…

- Very few tasks are wait-free solvable
  - ✓Most can only be solved detecting failures
  - ✓Set agreement, renaming, symmetry breaking
- Failure detectors: private oracles that give hints about failures




Bob's FD module is private!

# Weakest failure detectors

D is the weakest failure detector for a task T if

- Sufficient: D solves T
- Necessary: any D' that solves T implements D (provides at least as much information about failures as D does)

- consensus: $\Omega$ (the leader FD) [CHT96]
- set agreement: anti-$\Omega$ [Zie07]
- k-set agreement: anti-$\Omega_k$ [GK09]

# Progress with failure detectors

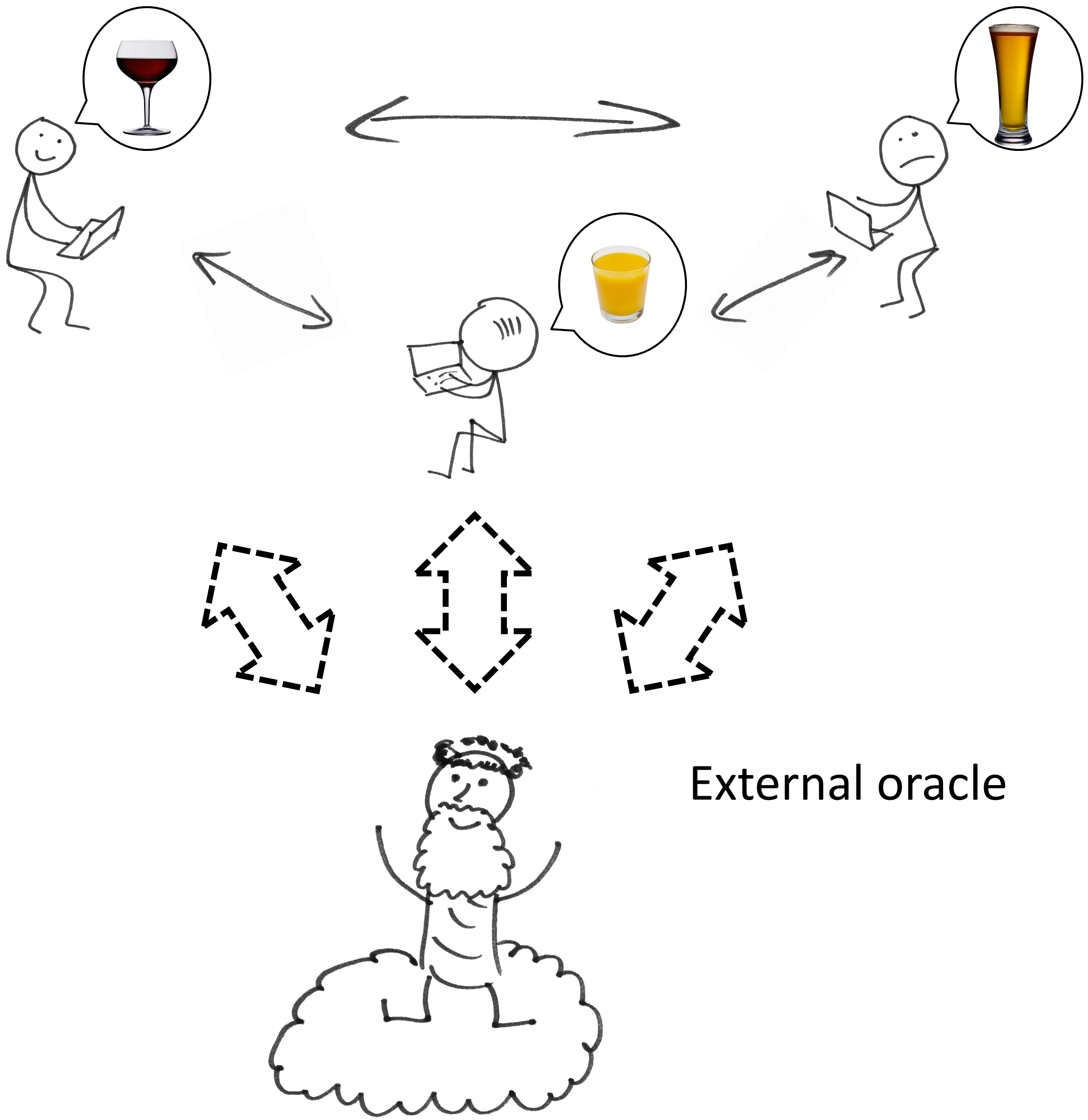
Assuming that every correct process takes enough steps

- Every correct process outputs
  - ✓Individual progress depends on other processes

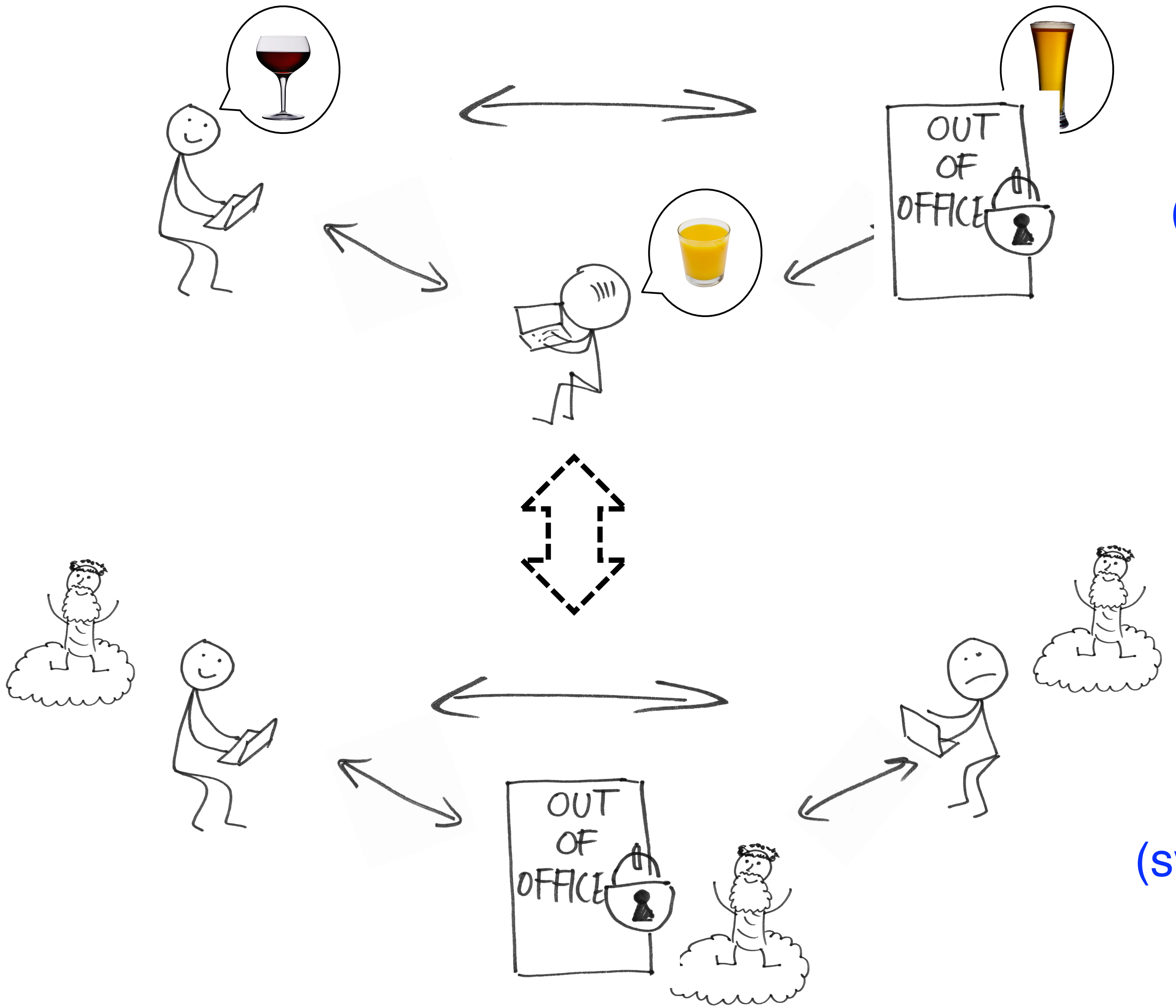
But can we solve a “hard” task wait-free?

External oracle: wait-freedom with advice

# External oracles



# External failure detection



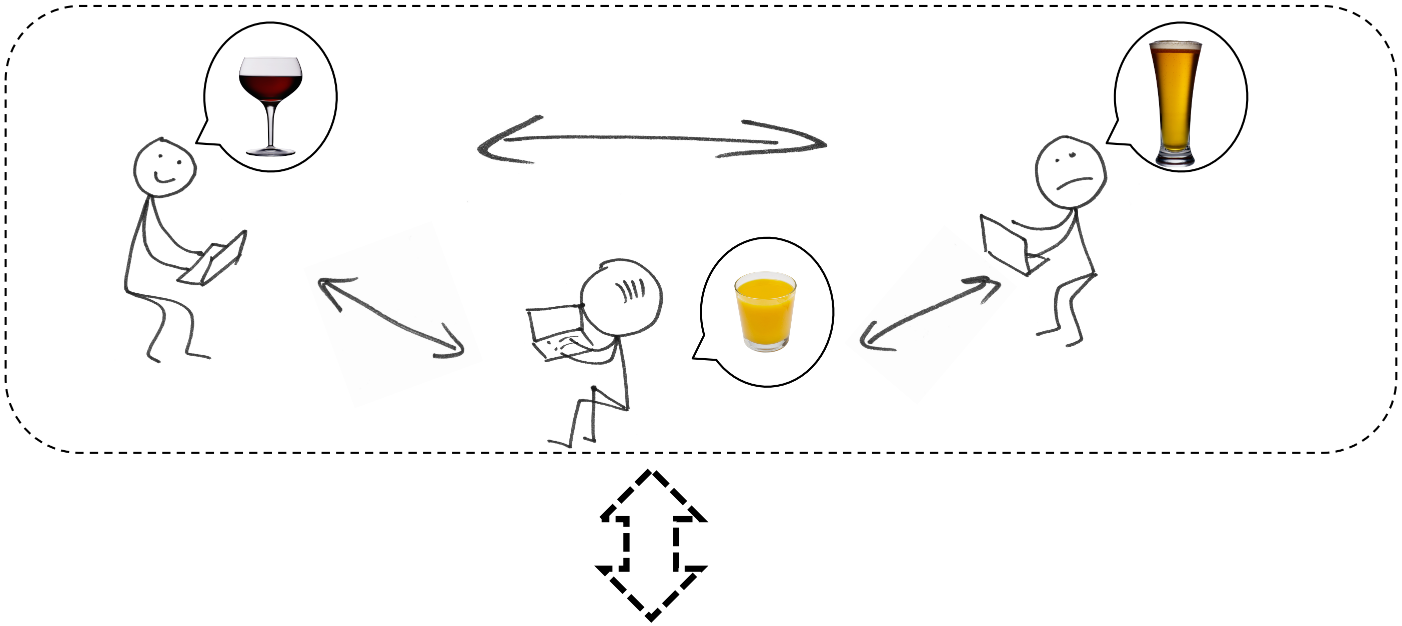

C-processes
(computation)

S-processes
(synchronization)

# Wait-freedom with advice

Assuming that every correct synchronization process takes enough steps

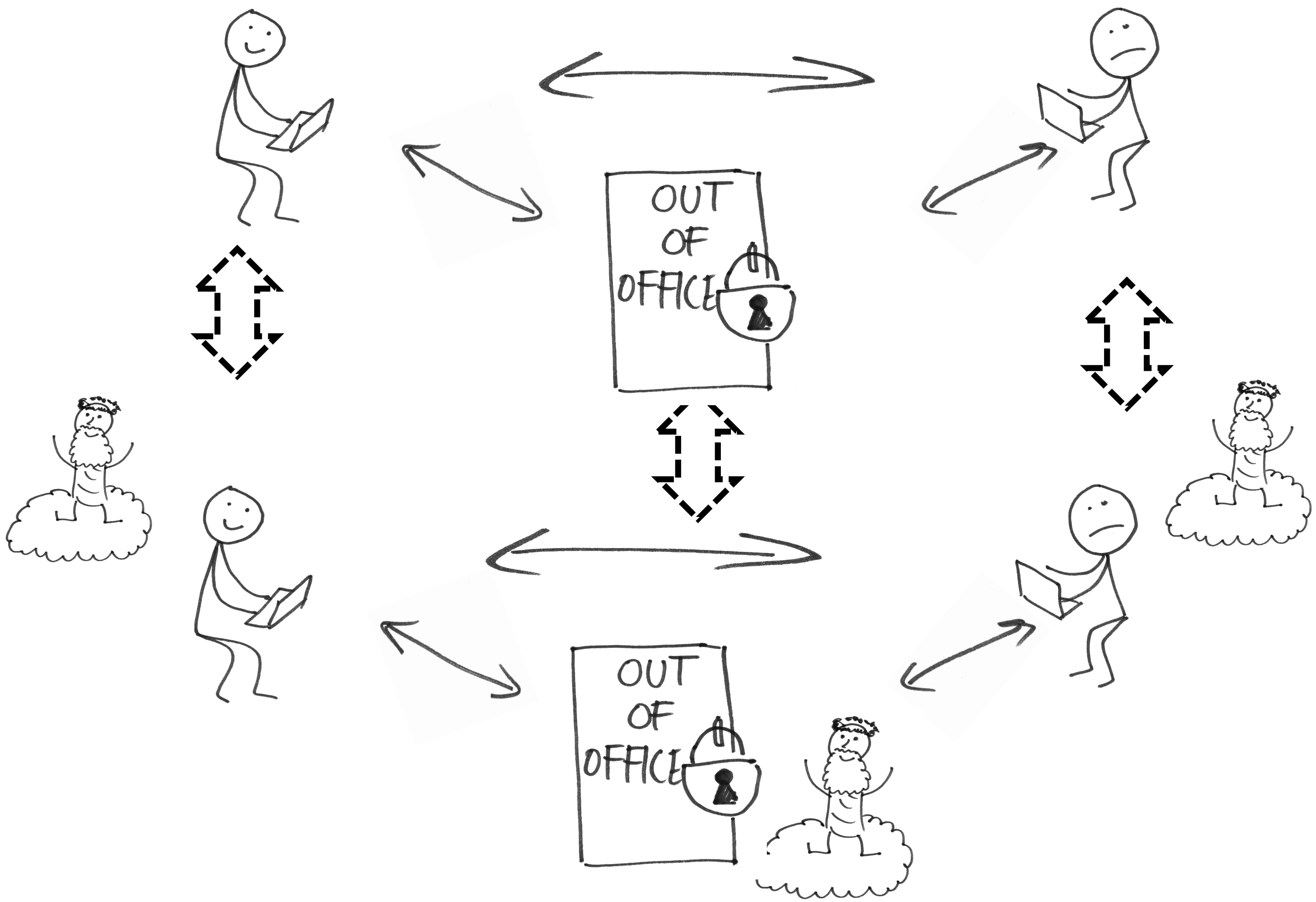
- Each computation process taking enough steps outputs
  - ✓Wait-freedom for C-processes



...

# EFD vs. FD

- Conventional (FD) model is a special case of EFD
- In EFD, the weakest failure detector for T  is at least as strong



# Special case: colorless tasks

- EFD and FD are equivalent w.r.t. colorless tasks:
  - ✓D solves a colorless T iff it solves T in EFD
  - ✓Weakest FDs for T are the same in the two models

What about generic (colored) tasks?

# A task characterization: k-concurrency

- Every task T is characterized by its concurrency level:
  - ✓The largest k such that T can be solved k-concurrently (assuming at most k participants run concurrently)
  - ✓k≥1 (every task is solvable 1-concurrently)
  - ✓n-concurrent solvability = wait-freedom
  - ✓k-set agreement has concurrency level k

# A task characterization

- k-concurrency can be simulated with anti-$\Omega_k$
  - ✓ A k-concurrently solvable task is solvable with anti-$\Omega_k$ (in EFD) [GG11,this paper]
- Each task is equivalent to some form of set agreement:
  - ✓ The WFD for every task of concurrency level k is anti-$\Omega_k$

# A hierarchy of n-process tasks

concurrency level

1 — Consensus ← universal tasks - $\Omega$

•
•
•

k — k-set agreement ← The easiest k-concurrent tasks anti-$\Omega_k$

•
•
•

n-1 — (n-1)-set agreement ← The easiest unsolvable tasks - anti-$\Omega$

n — n-set agreement ← Trivial tasks - no FD needed

# Implication: renaming

- (j,m)-renaming: j participants coming out with names in {1,…,m}
  - ✓In the conventional FD model, the problem is a FD

- (j,j+k-1)-renaming: k-concurrently solvable
  - ✓A variation of wait-free solution of (j,2j-1)-renaming [Attyia et al,1990]
  - ✓Concurrency lower bound is k
  - ✓ What about (k+1)-concurrency?

# Strong renaming (k=1)

(j,j)-renaming:

- Strong j-renaming has concurrency level 1
  - ✓By reduction to 2-process consensus [EBG09]
- The WFD for strong j-renaming is Ω

1 | Consensus, strong j-renaming | ⟵ universal tasks - Ω

# Weak j-renaming (k=j-1)

(j,2j-2)-renaming:

- When j is prime power: concurrency level j-1
  - ✓(j,2j-2)-renaming impossible wait-free (j-concurrently) [CR, 2010]
  - ✓The WFD for (j,2j-1)-renaming is anti-$\Omega_{j-1}$

| | | | |
|---|---|---|---|
| j-1 | (j-1)-set agreement, weak j-renaming | ← | The easiest (j-1)-concurrent tasks: anti-$\Omega_{j-1}$ |

- When j is not prime power: (j,2j-2)-renaming solvable wait-free [CR, 2011], and thus with anti-$\Omega_j$: concurrency level j
  - ✓ Can we solve (j,2j-3)-renaming with anti-$\Omega_j$?
  - ✓ Concurrency level of (j,m)-renaming?

# Outcomes

- New EFD framework, separating computation from synchronization
  - ✓New understanding of what does it mean to solve a task (with a FD)
- Complete characterization of all n-process tasks, based on their concurrency levels 1,…,n
  - ✓Including colored ones, like renaming or k-set agreement among a subset of k+1 processes

“*Problems cannot be solved by the same level of thinking that created them*”

THANK YOU!

# EFD vs. FD

- Conventional (FD) model is a special case of EFD
  - ✓Bijection between C-processes and S-processes
  - ✓A C-process fails iff its S-process counterpart does

- In EFD, the weakest failure detector is at least as strong
  - ✓Should let a C-process decide even if its S-counterpart has failed

# Simulations

- t-resilience ≅ t+1-process wait-freedom [BG93,Gaf09]
- Synchronous set agreement time lower bound [Gaf98,GGP05]
- k-concurrency ≅ k-set consensus [GG10]
- Adversaries, disagreement power [DFGT10,GK10]

But these simulations are asynchronous, what if failure detectors are used?

# A puzzle

Solving consensus among every pair of processes (with a FD) is as hard as solving consensus among all [Delporte et al., JACM 2010]

What about k-set agreement?

In EFD:

If D solves k-set agreement among some set U of k +1 C-processes, then D solves k-set agreement among all C-processes

(simple simulation of processes in U)